Varunesh Goyal
140070006

Codes used for successful gate level simulation and Modelsim output

The codes used are :

1. Bomb.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.thepackage.all;
entity bomb is
        port (  X : in input_symbol; Yo : out output_symbol; clk,reset : in std_logic);
end entity bomb;
architecture Behave of bomb is
  signal current_state : fsm_state;
  begin
        process (X, clk, current_state)
          variable nstate : fsm_state;
          variable vY : output_symbol;
          begin
                nstate := current_state;      --set the default values
                vY := n;
                case current_state is
                  when phi =>
                        if (X=b) then
                          nstate := a;
                        end if;
                  when a =>
                        if (X=o) then
                          nstate := b;
                        end if;
                  when b =>
                        if (X=m) then
                          nstate := c;
                        end if;
                  when c =>
                        if (X=b) then
                          nstate := a;
                          vY := y;
                        end if;
                  when d => nstate := d;
                  when e => nstate := e;
                  when f => nstate := f;
                end case;

                if (reset = '1') then Yo <= n;
                else Yo <= vY;
                end if;
                if(clk'event and clk = '1') then
                     if(reset = '1') then
                       current_state <= phi;
                     else
                       current_state <= nstate;
                     end if;
                end if;
        end process;
end Behave;
```

2. Gun.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.thepackage.all;

entity gun is
        port (  X : in input_symbol;
                Yo : out output_symbol;
                clk,reset : in std_logic);
end entity gun;

architecture Behave of gun is
  signal current_state : fsm_state;
  begin
        process (X, clk, current_state)
          variable nstate : fsm_state;
          variable vY : output_symbol;
          begin
                nstate := current_state;      --set the default values
                vY := n;
                case current_state is
                  when phi =>
                        if (X=g) then
                          nstate := a;
                        end if;
                  when a =>
                        if (X=u) then
                          nstate := b;
                        end if;
                  when b =>
                        if (X=n) then
                          nstate := phi;
                          vY := y;
                        end if;
                  when c =>
                        nstate := c;
                  when d =>
                        nstate := d;
                  when e =>
                        nstate := e;
                  when f =>
                        nstate := f;
                end case;

                if (reset = '1') then            --if reset put n to Y else vY to Y
                  Yo <= n;
                else
                  Yo <= vY;
                end if;
                if(clk'event and clk = '1') then
                     if(reset = '1') then
                       current_state <= phi;
                     else
                       current_state <= nstate;
                     end if;
                end if;
        end process;
end Behave;
```

3. Knife.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.thepackage.all;
entity knife is
        port (  X : in input_symbol;
                Yo : out output_symbol;
                clk,reset : in std_logic);
end entity knife;

architecture Behave of knife is
  signal current_state : fsm_state;
  begin
        process (X, clk, current_state)
          variable nstate : fsm_state;
          variable vY : output_symbol;
          begin
                nstate := current_state;      --set the default values
                vY := n;
                case current_state is
                  when phi =>
                        if (X=k) then
                          nstate := a;
                        end if;
                  when a =>
                        if (X=n) then
                          nstate := b;
                        end if;
                  when b =>
                        if (X=i) then
                          nstate := c;
                        end if;
                  when c =>
                        if (X=f) then
                          nstate := d;
                        end if;
                  when d =>
                        if (X=e) then
                          nstate := phi;
                          vY := y;
                        end if;
                  when e =>
                        nstate := e;
                  when f =>
                        nstate := f;
                end case;
                if (reset = '1') then            --if reset put n to Y else vY to Y
                  Yo <= n;
                else Yo <= vY;
                end if;
                if(clk'event and clk = '1') then
                     if(reset = '1') then
                       current_state <= phi;
                     else
                       current_state <= nstate;
                     end if;
                end if;
        end process;
end Behave;
```

4. Terror.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.thepackage.all;
entity terror is
        port (  X : in input_symbol; Yo : out output_symbol;
                clk,reset : in std_logic);
end entity terror;
architecture Behave of terror is
  signal current_state : fsm_state;
  begin
        process (X, clk, current_state)
          variable nstate : fsm_state;
          variable vY : output_symbol;
          begin
                nstate := current_state;  vY := n;    --set the default values
                case current_state is
                  when phi =>
                        if (X=t) then
                          nstate := a;
                        end if;
                  when a =>
                        if (X=e) then
                          nstate := b;
                        end if;
                  when b =>
                        if (X=r) then
                          nstate := c;
                        end if;
                  when c =>
                        if (X=r) then
                          nstate := d;
                        end if;
                  when d =>
                        if (X=o) then
                          nstate := e;
                        end if;
                  when e =>
                        if (X=r) then
                          nstate := phi;
                          vY := y;
                        end if;
                  when f =>
                        nstate := f;
                end case;

                if (reset = '1') then            --if reset put n to Y else vY to Y
                  Yo <= n;
                else  Yo <= vY;
                end if;
                if(clk'event and clk = '1') then
                     if(reset = '1') then
                       current_state <= phi;
                     else
                       current_state <= nstate;
                     end if;
                end if;
        end process;
end Behave;
```

5. Thepackage.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
package thepackage is
        type fsm_state is ( phi, a, b, c, d, e, f);
        type input_symbol is (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,space);
        type output_symbol is (n,y);

        component bomb is
                port (  X : in input_symbol;
                        Yo : out output_symbol;
                        clk,reset : in std_logic);
        end component bomb;

        component knife is
                port (  X : in input_symbol;
                        Yo : out output_symbol;
                        clk,reset : in std_logic);
        end component knife;

        component gun is
                port (  X : in input_symbol;
                        Yo : out output_symbol;
                        clk,reset : in std_logic);
        end component gun;

        component terror is
                port (  X : in input_symbol;
                        Yo : out output_symbol;
                        clk,reset : in std_logic);
        end component terror;

        component StringRecognizer is
                port (X : in std_logic_vector(4 downto 0); W: out std_logic;
                        clk, rreset: in std_logic);
        end component StringRecognizer;

        function BV_To_Input_Symbol(XY: std_logic_vector)  return input_symbol;
        function BV_To_Output_Symbol(X: std_logic_vector)  return output_symbol;
        function Output_Symbol_To_BV(K: output_symbol) return std_logic;
end thepackage;

package body thepackage is
        function BV_To_Input_Symbol(XY: std_logic_vector)  return input_symbol  is
                variable ret_var: input_symbol;
                begin
                if ( XY = "00001" )  then
                        ret_var := a;
                elsif (XY = "00010") then
                        ret_var := b;
                elsif (XY = "00011") then
                        ret_var := c;
                elsif (XY = "00100") then
                        ret_var := d;
                elsif (XY = "00101") then
                        ret_var := e;
                elsif (XY = "00110") then
                        ret_var := f;
                elsif (XY = "00111") then
                        ret_var := g;
                elsif (XY = "01000") then
```

```
                        ret_var := h;
                elsif (XY = "01001") then
                        ret_var := i;
                elsif (XY = "01010") then
                        ret_var := j;
                elsif (XY = "01011") then
                        ret_var := k;
                elsif (XY = "01100") then
                        ret_var := l;
                elsif (XY = "01101") then
                        ret_var := m;
                elsif (XY = "01110") then
                        ret_var := n;
                elsif (XY = "01111") then
                        ret_var := o;
                elsif (XY = "10000") then
                        ret_var := p;
                elsif (XY = "10001") then
                        ret_var := q;
                elsif (XY = "10010") then
                        ret_var := r;
                elsif (XY = "10011") then
                        ret_var := s;
                elsif (XY = "10100") then
                        ret_var := t;
                elsif (XY = "10101") then
                        ret_var := u;
                elsif (XY = "10110") then
                        ret_var := v;
                elsif (XY = "10111") then
                        ret_var := w;
                elsif (XY = "11000") then
                        ret_var := x;
                elsif (XY = "11001") then
                        ret_var := y;
                elsif (XY = "11010") then
                        ret_var := z;
                elsif (XY = "11011") then
                        ret_var := space;
                else  ret_var := space;
                end if;
                return(ret_var);
        end BV_To_Input_Symbol;

        function BV_To_Output_Symbol(X: std_logic_vector)  return output_symbol  is
                variable ret_var: output_symbol;
                begin
                        if (X = "0") then
                          ret_var := n;
                        else
                          ret_var := y;
                        end if;
                return(ret_var);
        end BV_To_Output_Symbol;

        function Output_Symbol_To_BV(K: output_symbol) return std_logic is
                begin
                        if(K = n) then return('0');
                        else return('1');
                        end if;
        end Output_Symbol_To_BV;
end package body;
```

6. StringRecognizer.vhd :

```vhdl
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.thepackage.all;

entity StringRecognizer is
        port (X : in std_logic_vector(4 downto 0); W: out std_logic;
                clk, rreset: in std_logic);
end entity StringRecognizer;

architecture Behave of StringRecognizer is
        signal abstractin : input_symbol;
        signal ay1, ay2, ay3, ay4 : output_symbol;
        signal y1, y2, y3, y4 : std_logic;
begin
        abstractin <= BV_To_Input_Symbol(X);

        fsm_bomb : bomb port map(X=>abstractin, Yo=>ay1, clk=>clk, reset=> rreset);
        fsm_gun : gun port map(X=>abstractin, Yo=>ay2, clk=>clk, reset=> rreset);
        fsm_knife : knife port map(X=>abstractin, Yo=>ay3, clk=>clk, reset=> rreset);
        fsm_terror : terror port map(X=>abstractin, Yo=>ay4, clk=>clk, reset=> rreset);

        y1 <= Output_Symbol_To_BV(ay1);
        y2 <= Output_Symbol_To_BV(ay2);
        y3 <= Output_Symbol_To_BV(ay3);
        y4 <= Output_Symbol_To_BV(ay4);

        W <= y1 or y2 or y3 or y4;

end Behave;
```

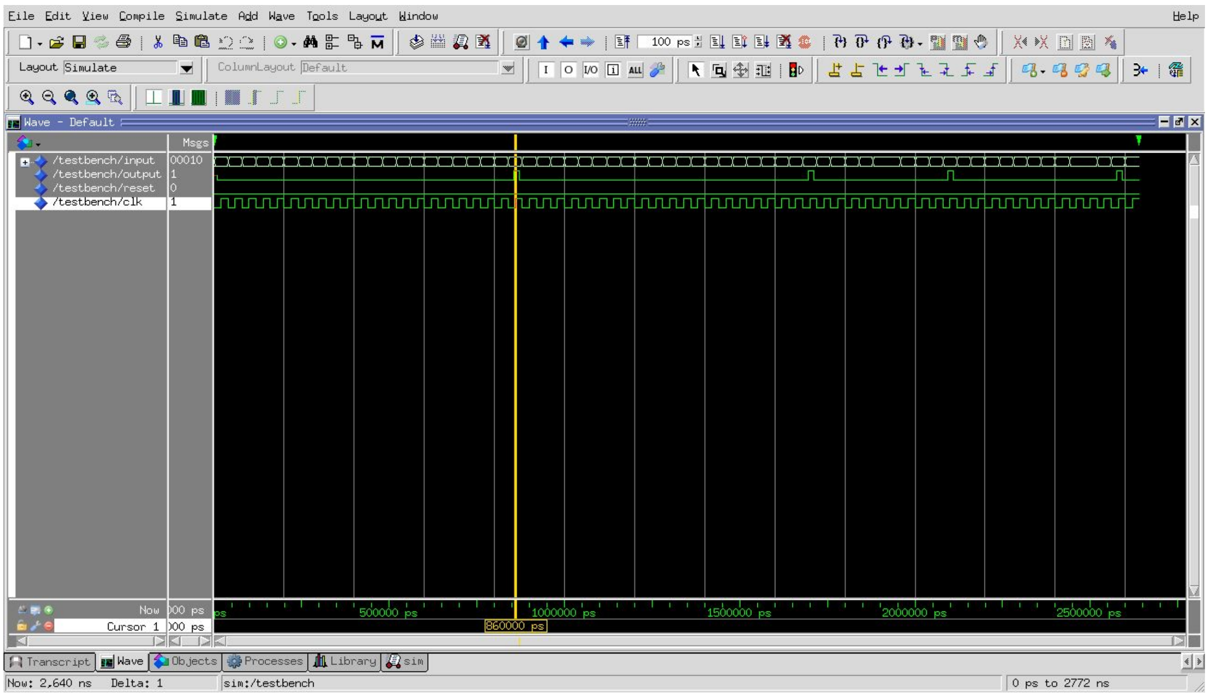
7. Testbench.vhd :

```
library ieee;
use ieee.std_logic_1164.all;
library std;
use std.textio.all;

entity Testbench is
end entity;

architecture Behave of Testbench is
  signal input : std_logic_vector(4 downto 0);
  signal output : std_logic;
  signal reset: std_logic := '0';
  signal clk: std_logic := '0';

  component StringRecognizer is
              port (X : in std_logic_vector(4 downto 0); W: out std_logic;
                      clk, rreset: in std_logic);
  end component StringRecognizer;

  function to_string(x: string) return string is
      variable ret_val: string(1 to x'length);
      alias lx : string (1 to x'length) is x;
        begin
          ret_val := lx;
      return(ret_val);
  end to_string;

  function to_std_logic (x: bit) return std_logic is
  begin
        if(x = '1') then return ('1');
        else return('0'); end if;
  end to_std_logic;

  function to_std_logic_vector (x: bit_vector) return std_logic_vector is
        variable y : std_logic_vector((x'length-1) downto 0);
        begin
        for k in 0 to (x'length-1) loop
                if(x(k)='1') then y(k) := '1';
                else y(k) := '0';
                end if;
        end loop;
        return y;
  end to_std_logic_vector;

begin

  dut: StringRecognizer port map(X => input , W => output, clk => clk, rreset => reset);

  process
    variable err_flag : boolean := false;
    File INFILE: text open read_mode is "TRACEFILE.txt";
    FILE OUTFILE: text  open write_mode is "OUTPUTS.txt";
    variable input_bv: bit_vector (4 downto 0);
    variable expected_output: bit;

    variable INPUT_LINE: Line;
    variable OUTPUT_LINE: Line;
    variable LINE_COUNT: integer := 0;

    variable clk_bit: bit;
```

```
        variable reset_bit: bit;

        begin

            while not endfile(INFILE) loop

              -- clock = '0', inputs should be changed here.
              LINE_COUNT := LINE_COUNT + 1;
              readLine (INFILE, INPUT_LINE);
              read (INPUT_LINE, clk_bit);
              read (INPUT_LINE, reset_bit);
              read (INPUT_LINE, input_bv);
              read (INPUT_LINE, expected_output);

              clk <= to_std_logic(clk_bit);
              input <= to_std_logic_vector(input_bv);
              reset <= to_std_logic(reset_bit);

              wait for 20 ns;

              -- check Mealy machine output and
              -- compare with expected.
              if (not (output = to_std_logic(expected_output))) then
                 write(OUTPUT_LINE,to_string("ERROR: line "));
                 write(OUTPUT_LINE, LINE_COUNT);
                 writeline(OUTFILE, OUTPUT_LINE);
                 err_flag := true;
              end if;

              if(endfile(INFILE)) then
                    exit;
              end if;

              -- clk = '1', inputs should not change here.
              LINE_COUNT := LINE_COUNT + 1;
              readLine (INFILE, INPUT_LINE);
              read (INPUT_LINE, clk_bit);
              read (INPUT_LINE, reset_bit);
              read (INPUT_LINE, input_bv);
              read (INPUT_LINE, expected_output);
              clk <= to_std_logic(clk_bit);
              input <= to_std_logic_vector(input_bv);
              reset <= to_std_logic(reset_bit);

              wait for 20 ns;

            end loop;

            assert (err_flag) report "SUCCESS, all tests passed." severity note;
            assert (not err_flag) report "FAILURE, some tests failed." severity error;

            wait;
      end process;

    end Behave;
```

As seen from the delay provided in the testbench, the given implementation of the FSM works perfectly with a clock of time period 40ns.

The Modelsim output wave :



```
/testbench/input   00010
/testbench/output  1
/testbench/reset   0
/testbench/clk     1
```

```
Now: 2,640 ns    Delta: 1    sim:/testbench    0 ps to 2772 ns
```